UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 1586

Parallelism in Event-Based Computations with Applications in Biology

PAVOL BAUER

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2017

ISSN 1651-6214
ISBN 978-91-513-0125-9
urn:nbn:se:uu:diva-332009

Dissertation presented at Uppsala University to be publicly examined in 2347, ITC, Lägerhyddsvägen 2, Uppsala, Monday, 11 December 2017 at 10:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Markos A. Katsoulakis (University of Massachusetts, Department of Mathematics and Statistics).

Abstract

Bauer, P. 2017. Parallelism in Event-Based Computations with Applications in Biology. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 1586. 48 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-0125-9.

Event-based models find frequent usage in fields such as computational physics and biology as they may contain both continuous and discrete state variables and may incorporate both deterministic and stochastic state transitions. If the state transitions are stochastic, computer-generated random numbers are used to obtain the model solution. This type of event-based computations is also known as Monte-Carlo simulation.

In this thesis, I study different approaches to execute event-based computations on parallel computers. This ultimately allows users to retrieve their simulation results in a fraction of the original computation time. As system sizes grow continuously or models have to be simulated at longer time scales, this is a necessary approach for current computational tasks.

More specifically, I propose several ways to asynchronously simulate such models on parallel shared-memory computers, for example using parallel discrete-event simulation or task-based computing. The particular event-based models studied herein find applications in systems biology, computational epidemiology and computational neuroscience.

In the presented studies, the proposed methods allow for high efficiency of the parallel simulation, typically scaling well with the number of used computer cores. As the scaling typically depends on individual model properties, the studies also investigate which quantities have the greatest impact on the simulation performance.

Finally, the presented studies include other insights into event-based computations, such as methods how to estimate parameter sensitivity in stochastic models and how to simulate models that include both deterministic and stochastic state transitions.

Keywords: Event-based computations, Parallel algorithms, Discrete-event simulation, Monte-Carlo methods, Systems biology.

Pavol Bauer, Department of Information Technology, Division of Scientific Computing, Box 337, Uppsala University, SE-75105 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-513-0125-9
urn:nbn:se:uu:diva-332009 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-332009)

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I P. Bauer, S. Engblom. Sensitivity estimation and inverse problems in spatial stochastic models of chemical kinetics. In *Lecture Notes in Computational Science and Engineering*, 2015.

II P. Bauer, S. Engblom, S. Widgren. Fast event-based epidemiological simulations on national scales. In *The International Journal of High Performance Computing Applications*. 2016.

III P. Bauer, J. Lindén, S. Engblom, B. Jonsson. Efficient inter-process synchronization for parallel discrete event simulation on multicores. In *ACM SIGSIM Principles of Advanced Discrete Simulation 2015*.

IV J. Lindén, P. Bauer, S. Engblom, B. Jonsson. Exposing inter-process information for efficient parallel discrete event simulation of spatial stochastic systems. In *ACM SIGSIM Principles of Advanced Discrete Simulation 2017*.

V P. Bauer, S. Engblom, S. Mikulovic, A. Senek; Multiscale modeling via split-step methods in neural firing. In *Mathematical and Computer Modeling of Dynamic Systems*, 2017.

Reprints were made with permission from the publishers.

Related Work

- A. Milias-Argeitis, S. Engblom, P. Bauer, M. Khammash. Stochastic focusing coupled with negative feedback enables robust regulation in biochemical reaction networks. In *Journal of the Royal Society Interface*, 2015.

- S. Widgren, S. Engblom, P. Bauer, J. Frössling, U. Emanuelson, A. Lindberg. Data-driven network modelling of disease transmission using complete population movement data: spread of VTEC O157 in Swedish cattle. In *BMC Veterinary Research*, 2016.

- K. Patra, DJ. Lyons, P. Bauer, M. Hilscher, R. Leão, K. Kullander, A role for solute carrier family 10 member 4, or vesicular aminergic-associated transporter, in structural remodelling and transmitter release at the mouse neuromuscular junction. In *European Journal of Neuroscience*, 2015.

- P. Bauer, S. Engblom. The URDME manual Version 1.3. *Available at Arxiv*, 2017.

- S. Widgren, P. Bauer, S. Engblom. SimInf: An R package for Data-driven Stochastic Disease Spread Simulations. *Available at Arxiv*, 2017.

Contents

1. Introduction

Event-based computations are frequently used in modeling and simulation of systems where variable changes happen abruptly and discontinuously, as for example due to random fluctuations of dependent quantities. Such random fluctuations can be observed in many physical systems, including the magnetization of atomic spins or the diffusion of a particle dissolved in a fluid. The nature of the fluctuations is often related to the microscopic behavior of the system that is typically unmeasurable to the observer, as for example thermodynamic processes. In a practical sense, this means that the stochastic model abstracts the dynamics and the parametrization of the underlying scale. Generally, such an approach may also be useful in modeling of complex systems that are not inherently stochastic but whose underlying dynamics are simply not fully understood.

The drawback of stochastic modeling, however, is that such systems are difficult to analyse and may not be analytically solvable. Despite that, numerical solution of such systems can typically be obtained using methods known as *samplers* or *Monte Carlo methods*. These methods simulate the stochastic processes that occur in the system using computer-generated pseudo random numbers; a technique that however may require significant computational effort for solution of realistic problems. As time is often a critical factor in scientific work, it is generally desirable to speed-up such simulations, allowing a modeler to retrieve the numerical results in a fraction of the original computation time.

However, as frequency scaling of computer performance came to an end in recent times, a significant speed-up is only achievable by means of parallel computing. Parallel scaling, however, relies on the assumption that the computational work can be divided into small tasks which the computer processes concurrently at the computational time of approximately a single task. While this assumption is often valid for, say, processing of independent services running in the background of an operating system, it is often not valid for numerical computations. This is especially the case for Monte Carlo methods, which are iterative algorithms where the state update of one iteration strictly depends on the state update of the previous iteration. Written in this sequential manner, the algorithm is hardly parallelizable as the dependencies prohibit to run single iterations concurrently.

The formulation of *parallel versions* of such numerical algorithms is the main topic of this thesis. Event-based models can be formulated quite differently, resulting in different model properties. These properties in turn may

significantly affect the performance of the parallel simulation. Thus I will mostly focus on models related to the Chemical Master Equation (CME) and the Reaction-Diffusion Master Equation (RDME). Characteristic properties of such models are that state transactions are governed by Poissonian processes, which act either locally or spatially extended via so-called short range interactions. The following thesis discusses how the solution algorithms of such models may be modified so that the algorithms become scalable on modern multicore computers. As the actual parallel efficiency of the modified algorithms is typically not ideal, it is also desirable to understand which factors limit the scalability.

The thesis is structured as follows: In §2 I give a brief overview of event-based modeling and present three specific event-based models that are applied in biology. In §3 I discuss how event-based models are simulated sequentially and how parameter sensitivities of event-based models can be estimated. In §4 I discuss how parallelism can be achieved in event-based models, and give a wider introduction to parallel discrete-event simulation. I also discuss more detailed topics in parallel sampling of RDME models, as for example how to obtain efficient synchronization while maintaining the correctness of the sequential model. Finally, I summarize the papers contributing to the thesis in §5 and give a conclusion of the work in §6.

2. Event-based models

In this chapter I will first give a brief introduction into event-based modeling in §2.1, followed by a description of three examples of event-based models in §2.2.1–2.2.3. The discussed examples are also used in the papers attached to this thesis.

2.1 Introduction

A variety of observable processes consist of continuous changes of an observed quantity. When such processes are modeled, they are typically formulated using algebraic or differential equations. However, when dependent variables change instantaneously and discontinuously, it may be easier to define the model in terms of so-called *events*. Each event then defines a discrete point in the dependent dimension, that is typically time, at which a change is made to the variable.

Discrete event modeling is typically applied to systems where the state space is a discrete set. Examples of discrete state models include queues, arrival processes, or finite-state automata [8]. Continuous models can be also *approximated* as discrete event systems if it eases the implementation or the analysis of the model behavior [43]. The model behavior itself is described by the sequence of many events. The sequence can be given by a predefined list, a set of logical rules, or a stochastic process.

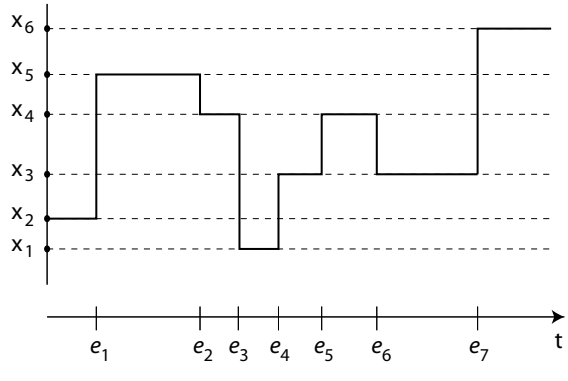

Figure 2.1. A realization of an one-dimensional random walk on a discrete state space.

A simple example of a discrete-event system is a *random walk* of a particle on a discrete plane in one dimension. The state of the system can be described by the position of the particle in the discrete set $\mathbb{X} = \{x_i : x_i \in \mathbb{Z}\}$. Events occurring at time $t_1 \dots t_j$ can be defined by the natural set $E = \{e_1 \dots e_j\}$. Furthermore, the model requires the definition of the initial state and some initial time. Figure 2.1 shows a possible outcome of the model.

2.2 Examples of event-based models

First, mesoscopic models of reacting and diffusing chemical species are discussed in §2.2.1. Next, event-based models of infectious spread on spatial networks are introduced in §2.2.2. Finally, models of spiking neuronal activity dependent on stochastic ion channel gating are explained in §2.2.3.

2.2.1 Models of chemical kinetics

Computational models of reacting and diffusing chemical species are an essential tool in the field of computational systems biology. Reaction and diffusion processes can be observed at different scales, where the physics of each scale requires a different mathematical model. The mesoscopic scale lies between the microscopic scale, that is typically modeled using quantum dynamics theory, and the macroscopic scale, which is usually modeled using differential equations. The mesoscopic scale is considered as appropriate for system sizes of $10^1 - 10^6$ individual molecules.

Chemical Master Equation

The dynamics on the mesoscopic scale are described by the Reaction Diffusion Master Equation (RDME), which is the spatial extension of the Chemical Master Equation (CME) [21, 32]. The CME describes reaction kinetics of some chemical species $j = 1 \dots D$ whose copy numbers enter the discrete random vector $\mathbb{X} = \mathbb{X}(t), \mathbb{X} \in \mathbb{Z}_+^D$, and $r = 1 \dots R$ reactions which cause transitions between states as

$$\mathbb{X} \xrightarrow{w_r(\mathbb{X})} \mathbb{X} - \mathbb{N}_r. \tag{2.1}$$

Here, $\mathbb{N} \in \mathbb{Z}_+^{D \times R}$ is a stochiometry matrix denoting the discrete change in the state due to a reaction and $w_r(x)$ is the propensity function of the reaction given by

$$w_r(\mathbb{X}) = \lim_{\Delta t \to 0} \frac{P(\mathbb{X} - \mathbb{N}_r, t + \Delta t) - P(\mathbb{X}, t)}{\Delta t}. \tag{2.2}$$

The propensity function can be understood as the probability that the reaction will occur in an infinitesimal interval $[t, t + \Delta t]$. As a state $\mathbb{X} - \mathbb{N}_r$ at time $t + \Delta t$ is solely dependent on the previous state $\mathbb{X}$ at time t, $\mathbb{X}$ is a *Markov process*.

It is possible to explicitly evolve the probability density function of such a system using the forward Kolmogorov equation or CME,

$$\begin{aligned}\frac{\partial p(\mathbb{X},t)}{\partial t} &= \sum_{r=1}^{R} w_r(\mathbb{X}+\mathbb{N}_r)p(\mathbb{X}+\mathbb{N}_r,t) - \sum_{r=1}^{R} w_r(\mathbb{X})p(\mathbb{X},t) \\ &=: \mathscr{M}\, p, \end{aligned} \tag{2.3}$$

where $p(\mathbb{X},t) := P(\mathbb{X} = \mathbb{X}(t)|\mathbb{X}(0))$ for brevity.

Due to the curse of dimensionality, equation (2.3) can be solved analytically only for simple models involving a small number of chemical species. The solution of more complex models thus relies on sampling methods, which are discussed in §3. Such samplers generate single trajectories of the stochastic system, but the histogram of many such realizations converges to (2.3).

We may also formulate a trajectory-wise representation of the CME probability density, using the *random time change representation* (RTC) stipulated by Kurtz [18]. In this representation, the state $\mathbb{X}_t$ is given as a sum of R independent unit-rate Poisson processes Π_r, representing the reactions that occurred the initial time $t = 0$. The RTC is written as,

$$\mathbb{X}_t = \mathbb{X}_0 - \sum_{r=1}^{R} \mathbb{N}_r \Pi_r \left(\int_0^t w_r(\mathbb{X}_{t-})\, dt \right), \tag{2.4}$$

where $\mathbb{X}_0$ is the initial state and $\mathbb{X}_{t-}$ denotes a state before the occurrence of any transitions at time t. Alternatively, one may construct the random counting measure $\mu_r(dt) = \mu_r(w_r(\mathbb{X}_{t-}); dt)$, that is associated with the Poisson process for the rth reaction at rate $w_r(\mathbb{X}_{t-})$ for any time t [16]. The RTC in (2.4) can be then written as the stochastic differential equation

$$d\mathbb{X}_t = -\mathbb{N}\mu(dt), \tag{2.5}$$

where $\mu = [\mu_1, \ldots, \mu_R]^T$ is the counting measures in vector form.

An exemplary model

An interesting CME model is the *stochastic focusing* system proposed by Paulsson et al. [45]. Consider two chemical species, an enzyme E and an intermediate complex C, that produce a product P at rate ν in the reaction

$$C + E \xrightarrow{\nu CE} E + P. \tag{2.6}$$

Furthermore, combine (2.6) with so-called birth and death transitions

$$\emptyset \underset{\beta_E E}{\overset{\alpha_E}{\rightleftharpoons}} E, \qquad \emptyset \underset{\beta_C C}{\overset{\alpha_C}{\rightleftharpoons}} C, \qquad P \xrightarrow{\beta_P P} \emptyset. \tag{2.7}$$

where molecules appear or disappear from and into the empty state $\emptyset$ at some birth rates α_C, α_E and decay rates $\beta_C, \beta_E, \beta_P$, respectively. At the macroscopic

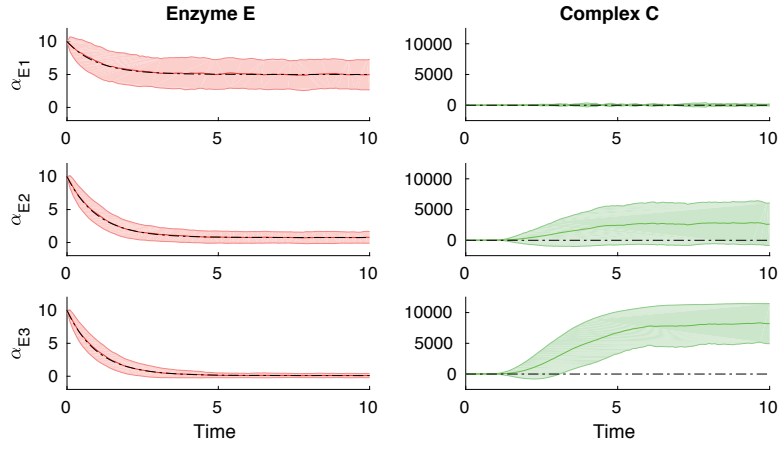

Figure 2.2. Solution of the model (2.7) for three different choices of the birth-rate α_E, either solved as deterministic rate equations (black dotted lines) or as the stochastic variant given by the CME (shadowed area, showing mean $\pm$ standard deviation).

scale, this system is typically modeled by ordinary differential equations, the so called *rate equations*. For this particular system, the rate equations are written as,

$$\frac{dE}{dt} = \alpha_E - \beta_E E, \tag{2.8}$$

$$\frac{dC}{dt} = \alpha_C - \beta_C C - \nu C E, \tag{2.9}$$

$$\frac{dP}{dt} = \nu C E - \beta_P P. \tag{2.10}$$

If the population of the involved chemical species is large, the solution of the CME model typically converges to the solution of the rate equation model. However, if the population of the species is small, discrete random effects play a role and may lead to significant differences between the solution of the CME and the rate equation models. This is exemplified in Figure 2.2, where the solutions of the CME model in (2.6) and (2.7), as well as the rate equations model (2.8) – (2.10) are shown for three different settings of parameter α_E. While the mean of the CME solution for complex C is close to the solution of the rate equations for the first setting of α_E, the effect of "stochastic focusing" clearly increases the average copy number of the complex C in the CME solution, when the last setting of parameter α_E is applied.

Reaction-Diffusion Master Equation

As noted previously, the RDME can be regarded as a spatial extension of the CME. Such an extension is necessary, if molecules in the system can not be assumed to be "well-stirred", or uniformly distributed in the space. From the modeling perspective, this is clearly the case for systems as biological cells, where different sub-cellular components may contain different concentrations of a considered species.

To capture the spatial dependency of an RDME model, the spatial domain V needs to be divided into K non-overlapping voxels $V_1 \dots V_K$. The division may result in a structured [14] or an unstructured [17] computational grid, which is now represented by a state space $\mathbb{X}$ of size $D \times K$. The molecules are allowed to move or *diffuse* from a voxel V_k to another voxel in its proximity V_j, while the assumption is that they are "well-stirred" within each voxel $V_{1\dots K}$. This type of diffusion-transition for a species i can be written as

$$\mathbb{X}_{ik} \xrightarrow{q_{kj}\mathbb{X}_{ik}} \mathbb{X}_{ij}, \tag{2.11}$$

where q_{kj} is the mesoscopic diffusion rate.

Similarly to (2.3), we can now write the diffusion master equation as

$$\begin{aligned}\frac{\partial p(\mathbb{X},t)}{\partial t} = &\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{K} q_{kj}(\mathbb{X}_{ik}+\mathbb{M}_{kj,k})p(\mathbb{X}_{1\cdot},\dots,\mathbb{X}_{i\cdot}+\mathbb{M}_{kj},\dots,\mathbb{X}_{D\cdot},t)\\ &-q_{kj}\mathbb{X}_{ik}p(\mathbb{X},t) =: \mathscr{D}p(\mathbb{X},t),\end{aligned} \tag{2.12}$$

where $\mathbb{M}_{kj}$ is a transition vector that is zero except for $\mathbb{M}_{kj,k} = -\mathbb{M}_{kj,j} = 1$. Finally, we arrive at the RDME, which combines (2.3) and (2.12) to

$$\frac{\partial p(\mathbb{X},t)}{\partial t} = (\mathscr{M}+\mathscr{D})p(\mathbb{X},t). \tag{2.13}$$

There are certain assumptions for the validity of the RDME. An important parameter is the mesh size parameter h, which in the case of structured meshes is the side length of each voxel. An obvious requirement is that the mesh size must be taken small enough to resolve geometrical features of the domain. Furthermore, a requirement is that h is chosen so that

$$\rho^2 \ll h^2 \ll \sigma^2\tau_\Delta, \tag{2.14}$$

where ρ is the molecular radius, τ_Δ the average molecular survival time, and $\sigma^2/2$ the macroscopic diffusion. The lower bound is necessary to guarantee that reaction events within a cell can be properly localized, while the upper bound guarantees the the voxel can be regarded as well-stirred.

As for the CME, numerical solutions of the RDME can be obtained using Monte Carlo sampling. Some dedicated methods for the solution of the RDME are discussed in §3.2 of this thesis.

2.2.2 Models of infectious spread on networks

Another exemplary event-based model is the model of infectious disease spread of populations residing on a spatial network. The infection model belongs to the class of compartment-based epidemiological models, where participating individuals are grouped into D discrete health states, for example denoting healthy, sick, or recuperated individuals. Given the assumption that the count of individuals is low, we can view the model as a version of the CME.

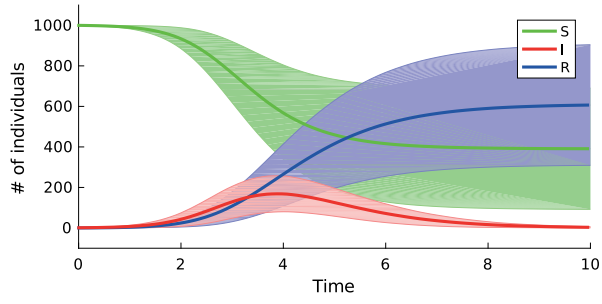

Figure 2.3. Solution to the SIR model in the stochastic representation.

We furthermore introduce discrete spatial locations denoted by the node index i, where the individuals are assumed to be uniformly distributed. As we now register the count of individuals contained in compartments $c = 1 \dots D$ at nodes $i = 1 \dots K$, the state space $\mathbb{X}$ becomes $\mathbb{Z}_+^{D \times K}$. We can now define the local dynamics analogously to (2.5) as

$$d\mathbb{X}_t^{(i)} = \mathbb{S}^{(i)} \mu(dt), \tag{2.15}$$

where $\mu(dt) = [\mu_1(dt), \dots, \mu_R(dt)]^T$ is again a vector of random counting measures for all R transitions, $\mu_k(dt) = \mu(R_k(X(t-)); dt)$, and $R : \mathbb{Z}_+^D \to \mathbb{R}_+^R$ is the transition intensities. The transition matrix $\mathbb{S}$ of size $\mathbb{Z}^{D \times R}$ is the negative of the stoichiometric matrix $\mathbb{N}$ in (2.1).

A prominent instance is the SIR-model [33], which categorizes individuals into susceptible, infected, or recovered compartments. The transitions of the SIR model are then,

$$S \xrightarrow{\delta SI} I, \qquad I \xrightarrow{\beta I} R, \tag{2.16}$$

where S, I, R are the susceptible, infected, and recovered compartments, and δ, β are the infection and recovery rates for the corresponding transitions. In our framework, the transition matrix of the SIR can be formulated as,

$$\mathbb{S} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}^T, \tag{2.17}$$

and the transition intensities are given as $R(x) = [\delta SI, \beta I]^T$. A numerical solution of the model in such a stochastic representation is shown in Figure 2.3.

We can further extend the model with interactions over a network, where the arcs of an undirected graph $\mathscr{G}$ define the connectivity between nodes. Such extensions may be useful to, for example, model movements of individuals over the network. Each node i may now affect the state of the nodes in its connected components $j \in C(i)$, and may be also be affected by other nodes j, where $i \in C(j)$. The network dynamics is then given by

$$d\mathbb{X}_t^{(i)} = -\sum_{j \in C(i)} \mathbb{C}\nu^{(i,j)}(dt) + \sum_{j;i \in C(j)} \mathbb{C}\nu^{(j,i)}(dt), \tag{2.18}$$

where $\nu^{(i,j)}$ and $\nu^{(j,i)}$ may be stochastic or deterministic counting measures.

Combining (2.15) and (2.18) we arrive at

$$d\mathbb{X}_t^{(i)} = \mathbb{S}\mu^{(i)}(dt) - \sum_{j \in C(i)} \mathbb{C}\nu^{(i,j)}(dt) + \sum_{j;i \in C(j)} \mathbb{C}\nu^{(j,i)}(dt), \tag{2.19}$$

which is the overall dynamics of the epidemiological model.

2.2.3 Multiscale modeling of firing neurons

An other exemplary event-based model related to the CME is the model of stochastic ion channel gating in spiking neurons. Spiking models describe how action potentials are initiated as the result of ion flux over neuronal membranes, and further propagate along the neuronal fiber. This flux is modulated by ion channels, a class of transmembrane proteins that are able to open and close microscopic pores in the membrane, allowing ions to follow the electrochemical gradient inward or outward of the intracellular space.

The gating process itself is depending either on the potential of the neuronal membrane or the binding of a ligand on the ion channel. In its microscopic formulation, the gating of an ion channel can be described by a kinetic scheme consisting of transitions between several closed states C_i and typically one open ion channel state O, as exemplified in Figure 2.4. The rates α and β are activation and deactivation rates of the channel, respectively. Clearly, such scheme can be directly translated into Markovian transition rules for the states $\{C_i, O\} \in \mathbb{Z}_+^8$, thus defining a continuous-time Markov chain $(t,s) \in \mathbb{R}_+ \times \mathbb{Z}_+^8$.

$$C_1 \underset{\beta}{\overset{3\alpha}{\rightleftharpoons}} C_2 \underset{2\beta}{\overset{2\alpha}{\rightleftharpoons}} C_3 \underset{3\beta}{\overset{\alpha}{\rightleftharpoons}} O$$

Figure 2.4. An exemplary gating scheme consisting of three closed and a single open ion channel state as well as three bidirectional transitions between them.

The total conductance of the ion channel G can be then computed from the fraction of open ion channels as,

$$G = \frac{\#O}{(\#O + \sum_i \#C_i)} \rho A \gamma, \tag{2.20}$$

where $\#O$ and $\#C_i$ is the discrete count of ion channel in the open or closed gate, respectively. Furthermore, ρ is the ion channel density, A the neuronal membrane area and γ the single channel conductance of the ion channel.

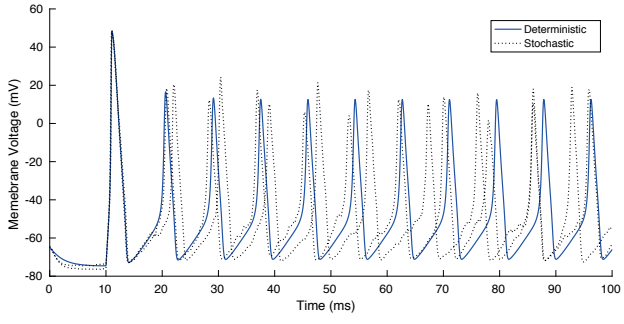

Figure 2.5. Numerical solution of a spiking neuronal model proposed by Hodking and Huxley [27] in the deterministic and stochastic representation. The firing is provoked by a current injection into an initial compartment of the neuronal structure.

We can than solve for the membrane potential V_m of the neuron affected by the ion channel by,

$$C_m \frac{d}{dt} V_m = G(V_m - E), \tag{2.21}$$

where C_m is the capacitance of the membrane and E is the membrane potential at which there is zero net flow over the boundary ("reverse-potential").

The model can thus be solved via a multi-level approach, where the ratio of open and closed channels in (2.20) is computed via a sampling method as discussed in §3, and the ordinary differential equation in (2.21) using, for example, the Crank-Nicholson scheme for time integration. An exemplary solution of stochastic ion channel gating in comparison to a deterministic representation is shown in Figure 2.5.

Note that the solution of (2.21) may require some additional variable coupling, as the activation and deactivation rates α, β in scheme 2.4 can be dependent on the membrane potential as $\alpha = \alpha(V_m), \beta = \beta(V_m)$. Likewise, considering a ligand-gated ion channel, the gating may be triggered by the firing of an adjacent neuron connected via a synapse, thereby making the rates dependent on the membrane potential of the presynaptic neuron.

3. Simulation of event-based models

In this chapter I will discuss the sequential numerical solution of event-based models. From a computer science perspective, this methodology is also known as Disrete-Event simulation, which is briefly discussed in §3.1. Next, I proceed with a discussion of simulation of some specific event-based models. The sampling of CME and RDME models is discussed in §3.2, with a special emphasis on the algorithmic efficiency of individual methods. Finally, I discuss methods for sensitivity estimation of CME models in §3.3.

3.1 Discrete-event simulation

In the computer-science literature, the numerical solution of event-based models is also known as discrete event simulation (DES). No uniquely defined DES algorithms exist, but one can agree that a typical discrete event simulator consists of three components; *a state*, *a clock* and *a scheduled event list* [8].

Before the simulation, the state is typically set to some initial values and the clock is set so some initial time. Furthermore the list of scheduled events is initiated with events, that will be processed according to the scheduled time. A DES algorithm in the simplest form then picks the next scheduled event from the event list, and processes it by updating the state and advancing the clock. In case the event generates one or more future future events, the new events are inserted into the scheduled event list. The simulation than proceeds with the iterative processing of the scheduled events, until no more events are left or the final simulation time is reached.

It is up to the individual simulator implementation to decide which particular data-structures are used for the event list, the clock and the state. Likewise, the state update routine, and the generation of new events may be constructed according to the requirements of the particular model. As shown in the following sections, it is also straightforward to adapt discrete-event simulators to Monte-Carlo samplers.

3.2 Simulation of models of chemical kinetics

A prominent sampling method for the solution of the CME is the *Stochastic Simulation Algorithm* (SSA), also known as Gillespie's Direct Method [23] or kinetic Monte-Carlo. As a consequence of the Markov property, the time τ_r

between two reaction transitions is an exponential random variable of the rate $1/w_r(\mathbb{X})$. The SSA uses inverse transform sampling to generate the time to *any* next reaction

$$\tau = -\lambda^{-1} \log u, \tag{3.1}$$

where $\lambda = \sum_r w_r(\mathbb{X})$ is the total reaction intensity and u is a uniformly distributed random number in the range $u \in (0,1)$. Next, it is decided which reaction r happened at the time $t+\tau$ by inverse sampling of a discrete distribution,

$$\sum_{s=1}^{r-1} w_s(\mathbb{X}) < \lambda u \leq \sum_{s=1}^{r} w_s(\mathbb{X}), \tag{3.2}$$

where u is another uniformly distributed random number. When the reaction index has been sampled, the state is updated according to (2.1), the time is set to $t = t+\tau$, and the algorithm proceeds iteratively with sampling of (3.1) and (3.2), until the final simulation time is reached.

The *Next Reaction Method* (NRM) [22] is essentially an optimization of the SSA, that is more efficient for the simulation of models consisting of a large number of chemical species or reactions channels. The state update of each SSA iteration is typically followed by a recomputation of all intensities $w_r(\mathbb{X})$, which are required for sampling of (3.1) and (3.2) in the consecutive iteration. In contrast to the plain SSA, the NRM recomputes only the propensities of the reactions that are dependent on the reaction which fired last, and which typically involves the same reactants or products.

The dependencies between the reaction channels are implemented via a dependency graph G, whose edges $G_{r,j}$ mark the reactions $j \in 1 \ldots J$ that are dependent on the firing of some reaction r. After the firing of reaction r, the next reaction time for each jth reaction is scaled as,

$$\tau_j^{\text{new}} = t + \left(\tau_j^{\text{old}} - t\right) \frac{w_j^{old}}{w_j^{new}}, \tag{3.3}$$

where τ_j^{old} and w_j^{old} is the reaction time and propensity rate of the jth reaction *before the state update* of reaction r, respectively. Note that the next reaction time for reaction r has to be sampled anew as $\tau_r^{\text{new}} = exp(1/w_r^{new})$.

The scaling (3.3) however, requires the explicit storage of all next reaction times $\tau_i, i \in 1 \ldots R$, which is efficiently implemented as a sorted priority queue. As a consequence, the NRM determines the next firing reaction by reading the smallest next-reaction time τ_i from top of the queue, and thus spares the computation of (3.2). If the priority queue is implemented via a binary heap, the NRM scales as $O(log_2 R)$ as opposite to $O(R)$ of the plain SSA. Also note, that the NRM clearly has the form of standard DES, where the priority queue is equivalent to the list of scheduled events.

Approximative CME sampling methods have also been proposed, as for example the *Tau-Leaping* scheme. Related to the random time change representation (2.4), the scheme can be written as,

$$\mathbb{X}_{t+\tau} = \mathbb{X}_t - \sum_{r=1}^{R} \mathbb{N}_r \Pi_r \left(w_r(\mathbb{X}_t)\, \tau \right), \tag{3.4}$$

where τ is the size of the "leap step" that is taken in time, similar to a step in a forward-Euler ODE integration scheme. The assumption in (3.4) however is, that the propensities $w_r(\mathbb{X}_t)$ do not change substantially in the interval $[t, t+\tau]$. If the assumption is invalidated, it may lead to an inaccurate approximation of the propensities, which may lead to an infeasible (negative) state $\mathbb{X}$ after the time step. Several adaptations of Tau-Leaping have been proposed, which aim to resolve the problem by computing a safe selection of τ before or after an iteration in time [6, 42].

Although Tau-Leaping is typically more efficient when propensities are large, the method is less efficient than the SSA for small values of τ. Partly this is because the generation of Poisson pseudo-random numbers is more expensive than the generation of exponentially distributed random numbers. For such cases, the proposed schemes typically switch the simulation regime to the SSA, until a larger value of τ can be determined.

Due to the existence of diffusion events between individual voxels, the dimensionality of RDME models is typically large. As stated in §3.2, the original SSA is typically an inefficient sampling method for such cases. Thus, the optimizations of the NRM have also been applied to the spatially extended case, resulting in the commonly used *Next Sub-volume Method* (NSM) [14, Supplementary Methods], shown in Algorithm 1.

Further approximative methods to simulate RDME models are spatial Tau-Leaping [31, 28], the Gillespie Multi-particle Method [48], or the Diffusive Finite State Projection [13]. Note, that although some approximative methods may be more efficient than the NSM under some circumstances, their solutions may not satisfy the statistical properties of exactly generated solutions of the RDME [31].

3.3 Parameter sensitivity estimation

The sensitivity of a parameter quantifies how much of the parameter uncertainity propagates to the uncertainity of the output, i.e. how responsive the model is with respect to that parameter. As reported by Gutenkunst et. al [24], published models in Systems Biology models often show "sloppy" spectra of parameter sensitivities, which make the model behavior effectively dependent on only a few "stiff" parameter combinations. Methods for sensitivity analysis are helpful to assess such properties and may thus contribute to the development of more robust and reliable models.

Algorithm 1 The next subvolume method (NSM)

1: Let $t = 0$ and set the state $\mathbb{X}$ to the initial number of molecules. Generate the dependency graph G. Initialize priority queue H. For all $j = 1 \ldots K$ voxels, compute the sum λ_j^r for all reaction propensities $w_r(\mathbb{X}_j)$ and the sum λ_j^d for all diffusion rates.
2: Generate the initial waiting times $\tau_j \simeq Exp(1/(\lambda_j^r + \lambda_j^d)$. Store the values in the priority queue H as ordered key-value pairs.
3: Remove the smallest time τ_j from H. Generate a pseudo-random variable u_1.
4: if $u_1(\lambda_j^r + \lambda_j^d) < \lambda_j^r$ then a reaction occured in voxel j. Find out which one it was using Equation (3.2).
5: if $u_1(\lambda_j^r + \lambda_j^d) \geq \lambda_j^r$ then a molecule diffused from voxel j. Sample Equation (3.2) to determine which species diffused to which neighbor.
6: Update the state of the system by setting $t = t + \tau$ and $\mathbb{X} = \mathbb{X} - \mathbb{N}_r$.
7: Draw a new exponential random number τ_j for the currently occurred event.
8: Update all rates marked as dependent to the current event in G, and recompute the next waiting time as $\tau_j^{\text{new}} = t + \left(\tau_j^{\text{old}} - t\right) \frac{(\lambda_j^r + \lambda_j^d)^{old}}{(\lambda_j^r + \lambda_j^d)^{new}}$.
9: Update H and repeat from step 3 until the final time T is reached.

In the stochastic setting, the goal of sensitivity estimation is to characterize the mean effect of a parameter θ on some function of interest $f(\mathbb{X}_t(\theta))$. The sensitivity of the function then computes as,

$$sens\{f(\mathbb{X}_t(c)), \delta\} = \frac{E[f(\mathbb{X}_t(c+\delta))] - E[f(\mathbb{X}_t(,c))]}{\delta}, \tag{3.5}$$

where c is a fixed value of parameter θ and δ is its (potentially small) perturbation. Note that for the case $\varepsilon \to 0$, equation (3.5) is the finite-difference approximation of the partial derivative of function $E[f(\mathbb{X}_t(\theta))]$ with respect to parameter θ.

For stochastic models of chemical kinetics, the expectations in (3.5) can be computed via the sample mean of many independent trajectories for $f(X_t(c+\delta))$ and $f(X_t(c))$. However, as $Var[f(\mathbb{X}_t(c+\delta))]$ and $Var[f(\mathbb{X}_t(c))]$ can be large in comparison to $E[f(\mathbb{X}_t(c+\delta))]$ and $E[f(\mathbb{X}_t(c))]$, the estimation of (3.5) may not be satisfying. In order to obtain more useful results, it is thus necessary to reduce the variance in the ensemble difference in (3.5) by increasing the coupling between the perturbed and unperturbed functions.

Rathinam and colleagues [47] proposed two methods to improve the coupling of individual perturbed and unperturbed trajectories; the *Common Random Number* (CRN) and the *Common Reaction Path* (CRP) methods. In essence, the methods are based on the fact that the generic SSA produces

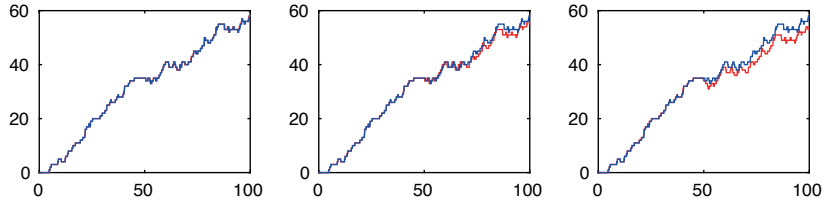

Figure 3.1. Evolution of a chemical species described by a unperturbed (blue) and perturbed (red) model solved with the CRP, where the size of the perturbation increases from the left to the right Figure [5].

trajectories from unspecified stochastic paths, as the random number generator (RNG) seed is typically not initiated prior to the sampling procedure. This already hints the trick of the CRN method, which amounts to specifying the same RNG seed for the sampling of the perturbed and unperturbed trajectories, leading to a stronger correlation between the solutions. The CRP method is a related method, but prescribes to (a) use individual streams of random numbers for each reaction channel $r \in 1 \dots R$, and (b) simulate the process with the NRM instead of the plain SSA. In the RTC representation (2.4) it can then be shown that the CRP generates coupled Poisson processes for perturbed and unperturbed reaction intensities, resulting in an even stronger correlation of in the solution of $X_t(c+\delta)$ and $X_t(c)$. For the coupled trajectories it can be then shown that,

$$E[\mathbb{X}_t(\theta+\delta;\omega) - \mathbb{X}_t(\theta;\omega)]^2 \sim O(\delta), \tag{3.6}$$

where ω now denotes that both trajectories are generated from the same stochastic path.

The application of the CRP to a simple birth-death model with one perturbed reaction rate is visualized in figure 3.1. In the leftmost Figure, it can be seen that the trend of the perturbed trajectory follows closely the fluctuations of the unperturbed one, although the copy number of the observed species is lower at the final simulation time in the perturbed solution. However, note that such correlation may vanish if the perturbation is too large or the process is observed at a longer time span.

Further methods for parameter sensitivity estimation in CME models include the Coupled Finite-Differences (CFD) [2], methods based on the Gyrshanov transform [46], or the Pathwise Information Theory [44]. While the CFD has been reported to reduce the variance stronger than the CRP for some CME models [50], its disadvantage is that perturbed trajectories have to be sampled coupled to the unperturbed ones, during the same simulation run. This renders the method unpractical for, e.g., variance reduction during numerical optimization, where the model is evaluated under many different input parameters.

4. Parallel simulation of event-based models

In this chapter I give an overview of parallel simulation of event-based models. First, I will present different approaches to parallelization in §4.1. Next, I disscuss synchronous space-parallel methods in §4.2 and asynchronous methods, also known as parallel discrete-event simulation, in §4.3. Finally, I discuss asynchronous sampling of RDME models in §4.4.

4.1 Approaches to parallelization

The general goal of parallelization is to divide some sequential workload into smaller parts, which can compute concurrently on several logical processes (LPs), which in practice may be physical or virtual computer threads or cores. *Weak scaling* is the ability of the parallel system to reduce the solution time with respect to the numbers of LPs for a fixed workload per LP, while *strong scaling* is the ability to reduce the solution time with respect to the number of LPs for a fixed total workload [25].

There are several approaches to parallel simulation of event-based models, which according to Liu [38] can be divided into four classes;

- *Replicated trials:* This type of parallelization prescribes to simulate multiple trials on concurrent LPs, as for example multiple realizations of a stochastic model. The parallel efficiency of this approach is typically high, as there is no requirement for synchronization between the simulating LPs. The approach is hence suitable for implementation on massively parallel infrastructures, as for example clusters or clouds, where trials can be distributed in a map-reduce fashion. However, the minimal parallel computation time is limited to the solution time of a single trial, i.e. the approach only scales weakly over a growing number of concurrent trials. Examples of framework for replicated trial simulation within the context of CME and RDME simulation are the *FastFlow* [1] or *MOLNs* [12] frameworks.

- *Functional decomposition:* This approach suggests that different parts of the event-based simulation algorithm used *within* a single simulator iteration can be parallelized on concurrent LPs. The scalability of the approach, in general, is expected to be low, as parts of a single iteration are typically strongly dependent on each other. Nonetheless, in specific models, it may be for example possible to parallelize the state update

routine. In the context of CME simulation, an approach which can be classified to this class is the concurrent sampling of subsets of reaction networks [41]. Such independent subsets can, for example, be found with help of a partitioning algorithm operating on the dependency graph of the reaction network. Clearly, this can be a successful strategy for parallel simulation of large reaction networks, which can be partitioned into several subsets.

- *Temporal decomposition:* This approach prescribes to divide the simulation time into smaller intervals, that are simulated concurrently on distinct LPs, followed by a global synchronization step. An example in the context of the CME is the application of the *Parareal* algorithm [37] to time-parallel simulation of such models, as discussed in [15]. This type of parallelization may scale strongly, but the efficiency is typically limited by the size of the concurrent time step and the frequency and duration of the global synchronization step.

- *Spatial decomposition:* In case the models are spatially extended, the spatial domain is divided into smaller subdomains, each of which is simulated concurrently, as visualized in Figure 4.1. Events occurring at the boundary of a subdomain must be processed by the neighboring LPs at the same simulation time, and any resulting conflicts must be resolved by means of some kind of synchronization. This type of parallelization has a large potential to scale weakly as well as strongly, but the actual efficiency heavily depends on the choice of the actual synchronization method and the nature of the simulated model.

Methods for simulation of spatially decomposed models can be further divided into synchronous and asynchronous methods. In *synchronous methods* the boundary events are synchronized at global timesteps, at which all LPs stop their local simulation process and communicate their boundary state to their neighbors. In *asynchronous methods*, the boundary conflicts are resolved interleaved with the local simulation, where neighboring LPs continuously exchange information of their boundary state. For event-based computations, such methods are also known as parallel discrete-event simulation (PDES).

4.2 Synchronous methods

As discussed previously, synchronous space-parallel methods synchronize their boundary state only at specific global times, that are known to all LPs. If the model is solved in discrete time and the step size is equal on each domain, the synchronization can be made after each such timestep. In the context of RDME models, such approach is applicable for e.g. Spatial Tau Leaping [31], where all reaction and diffusion processes advance at time $T_0, T_0 + \Delta t, T_0 +$

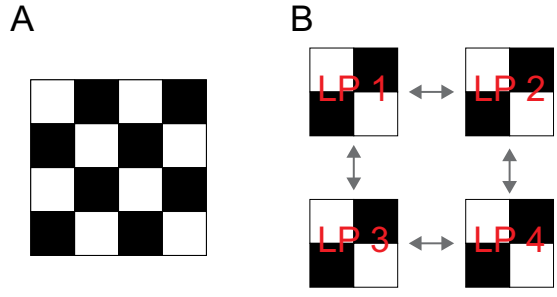

Figure 4.1. Spatial decomposition of a white-black colored Cartesian grid (A) into several subsets (B), where each of them is assigned to parallel simulation on individual LPs. The arrows between the subsets indicate the synchronization of border cells, which occurs after the propagation of each timestep. Adapted from [3].

$2\Delta t, \dots T_{end}$. In case the model is solved in continuous time, or the size of the discrete timestep differs on individual LPs, the method ignores the synchronization of the intermediate intervals $[T_0, T_0 + \Delta t], [T_0 + \Delta t, T_0 + 2\Delta t], \dots$ and continues synchronizing at the global timesteps $T_0, T_0 + \Delta t, T_0 + 2\Delta t, \dots T_{end}$ only. This typically incurs a numerical error in the solution of the model.

A framework for the analysis of the weak error in spatial stochastic models has been proposed by Arampatzis et al. [3]. Given that the model solves for some function of interest $f(\mathbb{X}_t)$ of the Markov process $\mathbb{X}_t$ the solution of the system can be written as an initial-value problem of the form,

$$u(t) = E(f(\mathbb{X}_t)). \tag{4.1}$$

We can further specify the generator $\mathscr{L}$ of the continuous-time Markov chain $\mathbb{X}_t$, whis relates to (4.1) as

$$\mathscr{L}u(t) = \partial_t u(t), \tag{4.2}$$

and the action of the Markov semigroup associated to generator $\mathscr{L}$ as

$$e^{t\mathscr{L}} = E(f(\mathbb{X}_t)). \tag{4.3}$$

Next, the authors propose to create two subsets of the domain with the index sets I^B and I^W, representing voxels that are colored in black or white, as exemplified in Figure 4.1. The generator $\mathscr{L}$ can be then decomposed as,

$$\mathscr{L} = \mathscr{L}^B + \mathscr{L}^W, \tag{4.4}$$

where $\mathscr{L}^B$ and $\mathscr{L}^W$ are the generators of subsets I^B and I^W, respectively. This enables to approximate the Markov semigroup based on the Trotter theorem [52], writing the propagator for each timestep Δt as,

$$e^{\Delta t\mathscr{L}} \approx e^{\Delta t\mathscr{L}^B} + e^{\Delta t\mathscr{L}^W}, \tag{4.5}$$

which is also known as the Lie splitting approximation. It can be shown that on a Cartesian grid the local error due to the the splitting (4.5) is then approximately $O(1/q)\Delta t^2$, where q is the size of the subdomains [3].

The spatial domain Ω is divided into K non-overlapping subdomains $\Omega_k, k \in 1 \dots K$, which are assigned to $LP_k, k \in 1 \dots K$, as shown in Figure 4.1. Each LP than solves the scheme in (4.5) for a single timestep Δt by (a) simulating all voxels in set I^B in parallel for time Δt using the SSA, (b) synchronizing boundary voxels in set I^B with boundary voxels in set I^W, (c) simulating all voxels in set I^W in parallel for time Δt using the SSA, (d) synchronizing boundary voxels in set I^W with boundary voxels in set I^B.

Due to the straightforward synchronization pattern, synchronous schemes as in (4.5) can be implemented on shared-memory computers, as well as on massively parallel systems, such as Graphical Computing Units (GPUs). The strongest impact on the parallel efficiency is then typically given by the size of the splitting time step, where a large timestep implies the best efficiency. Another synchronous approach for parallel simulation of RDME models can be found in [26]. Here the authors also discretize the continuous operator using Lie-Trotter splitting, and use the Diffusive Final-State Projection algorithm for the simulation on each subdomain.

4.3 Asynchronous methods

Asynchronous space-parallel methods, also known as Parallel Discrete-Event Simulation (PDES), resolve boundary conflicts asynchronously and interleaved with the local simulation of the affected LPs. Hence, such methods may be used for *exact parallel simulation* of spatially extended continuous-time models, or models where the size of the timestep differs on individual subdomains. The methods are typically implemented as concurrently running discrete-event simulators, whose structure has been discussed in §3.1 of this thesis. Each LP thus runs an instance of the simulator, containing a part of the model state, an event list containing all scheduled events, and advances its own simulation clock according to the list.

The main goal of asynchronous event processing in PDES is to implement the Virtual Time protocol [29]. Again, the spatial domain Ω is first divided into K smaller subdomains $\Omega_k, k \in 1 \dots K$ that are assigned to $LP_k, k \in 1 \dots K$ for local simulation. According to the Virtual Time protocol, each LP_k processes its local events $e_1^k, e_2^k, \dots e_n^k$ and thus advances its *local virtual time* LVT_k, defined as the timestamp of the last locally processed event at $\mathrm{LVT}_k = t(e_n^k)$. The *global virtual time* (GVT), on the other hand is defined globally as $\mathrm{GVT} = \min_{k \in \{1 \dots K\}} \mathrm{LVT}_k$, i.e. the minimum LVT of all LPs.

In order to resolve boundary conflicts, LPs are required to synchronize the state changes occurring at the boundary state by exchange of so-called time-stamped *messages*. Each message m_i^{jk} then contains the specification of the

event that was locally executed in LP_j and affects the remote state of LP_k, as well as the time of the message $t(m_i^{jk})$. LP_k then receives the message, enters it to its event list, and processes it in the right temporal order with the requirement that $t(e_{n-1}^k) \leq t(m_i^{jk}) \leq t(e_n^k)$, with respect to the l locally scheduled events $e_n^k, n \in \{1 \ldots l\}$ on LP_k. This means that all previously scheduled events with a smaller timestamp than the event in the message have to be processed beforehand, and all events with a greater timestamp have to be processed after the event contained in the message.

This is the *local causality constraint* (LCC) [19], which guarantees that a parallel simulator can generate exactly the same solution as in a sequential implementation. In practice, messages typically arrive out-of-order to the LVT of neighboring LPs, due to asynchrony in processing or communication latencies of the interconnect, thereby creating *causality errors*. This makes the implementation of the LCC a non-trivial task. To satisfy the LCC and avoid causality errors, two different classes of PDES methods have been proposed;

- In *conservative PDES* causality errors are strictly avoided. The connectivity of the LPs is given by a graph $\mathscr{G}$, whose arcs mark the dependencies between the assigned subdomains. When a local event e_n^k is processed on LP_k, it must be ensured that $t(e_n^k) < \mathrm{LVT}_j, \forall j \in C(k)$, where $C(k)$ is the strongly connected component of LP_k according to $\mathscr{G}$, i.e. all LPs that can send and recieve messages from and to LP_k. If this is not the case, LP_k blocks its execution and proceeds with the simulation only when the condition becomes true again. The "cost of lost opportunity" resulting from many such blocking periods typically are the main source of overhead in conservative PDES. As the approach is prone to deadlock, some sort of prevention mechanism has to be implemented, assuring that two LPs do not block simultaneously, waiting on each others simulation progress [9].

- In *optimistic PDES* causality errors are initially allowed, but must be resolved upon detection using so-called *rollbacks*. This means that upon arrival of a *straggler* message m_i with timestamp $t(m_i) < \mathrm{LVT}$ on the receiving LP, all n previously processed events $e_n, t(e_n) > t(m_i)$ have to be undone. This process is visualized in Figure 4.2. The management of previously processed events thus requires the implementation of an event history, which stores the information of all local events and messages that have been processed.
  As the rollback may also include previously received messages, the sending of those messages has to be undone as well, which may result in a *rollback cascade*, potentially affecting several LPs [19]. Importantly, the cascade may only affect processed messages with a timestamp greater than the GVT. Thus, all history items whose event time is smaller than

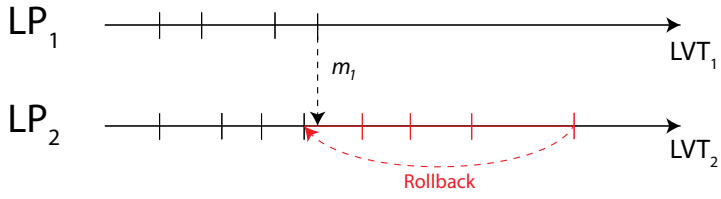

Figure 4.2. A snapshot of an asynchronous parallel simulation on two LPs. The local simulation time LVT_2 of LP_2 is higher than the local time LVT_1 of LP_1. The message m_1 sent from LP_1 at LVT_1 thus creates a rollback of LP_2, reverting all previously simulated events with a larger timestamp than $t(m_1) = \mathrm{LVT}_1$, shown in red color.

the GVT can be fossil-collected during runtime. To summarize, the main source of overhead in optimistic PDES is the cost of rollbacks.

It is difficult to state which class of PDES is more efficient for a given event-based model. Some intuition can be given based on the *lookahead* property of a model. The lookahead is defined as the average inter-arrival time between received messages, quantified on the receiving LP. Typically, it is suggested that models where the lookahead is larger than the average local inter-event time scale better using conservative PDES protocols. Note that models with such properties can also be parallelized by a central scheduler, that distributes the events onto LPs [56]. On the other hand, optimistic PDES protocols are recommended for event-based models where the lookahead is smaller or similar to the average local inter-event time. Optimistic methods are also preferred if the lookahead is a stochastic variable, or can not be quantified at all [11].

Furthermore, another class of *hybrid PDES* exists, which can be seen as a mixture of optimistic and conservative methods. The goal of hybrid methods is to balance the cost of rollbacks and the cost of lost opportunity by introducing optimism control.

4.3.1 Adaptivity

As discussed previously, optimistic methods are often preferred to conservative ones if no "lookahead" on future event arrival times is available. This is clearly the case for the exact numerical simulation of RDME models, where the time increments between events are exponentially distributed and hence unbounded from below [11].

Thus, *optimistic simulation* needs to be applied to this class of problems, where future events are executed speculatively, and causality errors are resolved using rollbacks. However, such methods are prone to "over-optimism", due to an overly large proportion of events that are processed speculatively. This may hinder the efficiency of the parallel simulation due to two main reasons.

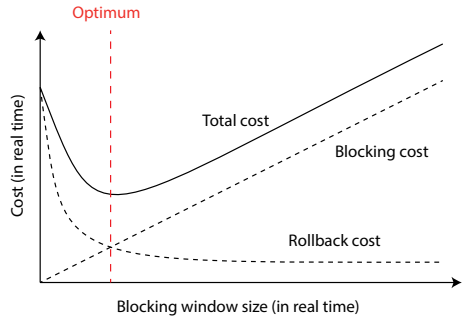

Figure 4.3. The relationship between the size of a static blocking window on the simulation cost of an optimistic parallel discrete-event simulator. Adapted from [10].

- First, when a causality error occurs, the local state must be rollbacked to the timestamp of the message that caused the error. Clearly, if the amount of speculatively executed events increases, the expected amount of rollbacked events increases, too. Hence, unnecessary speculative simulation will increase the total cost of rollbacks.

- Second, the amount of speculative execution also increases the probability of longer rollback cascades. As rollback cascades over several LPs pay an extraordinarily high cost to the overhead, they must be avoided in efficient simulations.

Therefore, *adaptive protocols* seek to limit the proportion of speculative processing that leads to a high probability of rollbacks. For best efficiency, such limitations should be imposed on the most speculative LPs, for example LPs that advance their LVT faster than their neighbors. The limitations may, for example, be implemented by a *blocking window*, during which the LP suspends its local simulation for a certain amount of real time s (e.g. in microseconds). The simplest approach is than to apply the blocking window at every increment of the LVT.

The window size s should be chosen for each LP so that the overall rollback cost of the simulation should be reduced, as visualized in Figure 4.3. However, as the blocking imposes a new source of overhead, the window size s is optimal when both the cost of rollbacks and the overhead due to the additional blocking time are minimized. In addition, the window size s should be chosen for each LP, so that all LPs advance their LVTs at an approximately equal rate throughout the simulation. Under this condition, the simulation is expected to be the most efficient as speculative events are less likely to lead to causality errors.

Different methods to compute and apply an adaptive blocking window have been proposed [10]. A relevant framework for optimism control based on an

adaptive blocking window size is the Elastic Time protocol [49]. Let us again define the graph $\mathcal{G}$, whose edges mark the existence of a bidirectional message channel between two LPs. The framework suggests to first compute an *error potential*, which is a measure of how far the LVT_k of LP_k is ahead of the LVTs of all LPs connected to LP_k. The error potential for LP_k writes as

$$\mathrm{EP}_k = max(\nu_k - \alpha_k, 0), \tag{4.6}$$

where ν_i is the timestamp of the local event that will be processed next on LP_k, or if the timestamp is temporally unknown, $\nu_k = \mathrm{LVT}_k$. Furthermore, $\alpha_k = min(\nu_j), \forall j \in C(k)$ is the *minimum future time* for all connected LPs, i.e. the time of the next event occurring on some LP in the connected component $C(k)$ of LP_k. Given a value of EP_k, a blocking window is determined as

$$\delta_k = \omega \cdot \mathrm{EP}_k, \tag{4.7}$$

where ω is a scaling factor, that is related to the average duration of a local state update in real time, and has to be hand-tuned for a specific model. The blocking window δ_k is than applied to LP_k, as shown in Algoritm 2. Note that in (4.6) we ignore the existence of transient messages, which are messages that are sent but not yet received, and thus have an impact on how EP_k is computed.

To summarize, the Elastic Time protocol applies the largest blocking window size to the LPs whose LVT is furthest away from the LVT of the slowest LP. Thus the algorithm fulfills the goal of adaptive methods, which is a simulation where all LPs advance their LVT at approximately the same rate.

However, the computation of variable α evidently requires frequent inter-LP exchange of the timestamp ν, which has to be available before the arrival of the actual future message on the receiving LP. For implementation in a distributed memory environment, the authors thus suggest to use a high-speed network dedicated for communication of ν. If the algorithm is implemented on shared-memory environment, timestamps ν may be simply read and written from a shared memory location.

Another important observation is that the protocol is only efficient for models where the future event time ν *is known or can be statistically estimated* by the sending LP. Without a reasonable estimate of ν, the speculative execution of connected LPs can not be effectively bound, hence an optimal trade-off as depicted in Figure 4.3 may not be attainable.

4.3.2 Scalability

It is difficult to make statements about the general scaling of space-parallel asynchronous methods. As discussed in §4.3.1, the scaling of a particular simulation depends strongly on the actual protocol maintaining the local causality constraint and reducing unnecessary overhead. Furthermore, *inherent model*

Algorithm 2 The Elastic Time protocol

1: Initialize the model state, set ν_i to the timestamp of the first locally scheduled event. Set δ_i to an initial value.
2: **while** the simulation is not complete **do**
3: Process the next local event and compute a new value for ν_i.
4: Start waiting timer τ.
5: **while** $\tau < \delta_i$ **do**
6: Receive messages m_i from other LPs. Go to line 10 in case they trigger rollbacks, or if some message timestamp $t(m_i) = \alpha_i$.
7: Update the state according to the messages and recompute ν_i.
8: Update the EP_i and δ_i according to (4.6) and (4.7), respectively.
9: **end while**
10: If necessary do rollbacks triggered by the messages, and recompute ν_i.
11: Save the state and carry out the fossil collection of the history list.
12: **end while**

properties can have a strong impact on the particular scalability of the parallel simulation. An incomplete list of such model properties includes;

- *Topology of the spatial network:* Simulations will in general scale better if spatial nodes are weakly connected, or their network consists of several smaller hubs that can be well separated from the overall network. On the other hand, models where spatial nodes are strongly interconnected are typically less scalable, due to a larger number of resulting inter-LP dependencies. According to Fujimoto [20], one of the main challenges in current PDES research is to allow for scalable simulation of irregular networks and networks with skewed node degree distributions.
- *Frequency of synchronization:* The scalability is typically better if the main part of the overall simulation work is done locally, as opposed to events where a high proportion of events affect several LPs and thus require a a frequent inter-LP synchronization.
- *Synchronization pattern:* It may have an impact on the scalability if events in the underlying model interact at close range only, or also distant nodes may be affected. Also, more than two LPs could potentially be involved in a single synchronization process, simultaneously.
- *Predictability of synchronization:* This property is related to the concept of "lookahead" discussed in §4.3. It has been shown [51] that if future messages are well predictable by receiving LPs, the scalability of the simulation is in general better, as opposed to models where future event updates are stochastic or unknown. This factor has also lead to the exploitation of the lookahead in specific event-based models [39].

These differences may explain the PDES performance results for models that scale over hundreds of thousands of LPs [4], to models that have been reported to be not scalable at all [11]. Unfortunately, a universal performance model integrating all inherent factors has not been yet proposed, leaving the study of PDES scalability to empirical measurements.

Furthermore, implementation details as well as hardware properties of the simulation environment can have a strong impact on the simulator performance. Asynchronous methods typically strongly depend on the memory management and efficiency of inter-LP communication. Thus, performance results for the same model may differ, depending on if the simulator has been implemented on multicores, [54], manycores [55], or virtual machines running in a cloud [40].

However, successful predictive modeling of PDES performance has been proposed for some specific types of event-based models. For example, Korniss and colleagues [36, 35] studied the evolution of the virtual time horizon (VTH) resulting from conservative PDES of Glauber spin-flip dynamics, a spatial stochastic system studied in the area of statistical physics. The VTH can be seen as a snapshot of the LVT of all LPs, taken at some point in real time, as visualized in Figure 4.4.

The spatial domain consists of regular lattices which contain spins, which can be in one of two states (+1 or -1). Given a system of L spins, connected in one dimension, and with reflecting boundary conditions at both ends of the chain. This domain is then mapped to L parallel LPs, where each LP carries one site, i.e. one spin, of the underlying system.

The update of each site is an independent Poisson process η_i with the same rate, that is independent of the state of the underlying spin. After the sampling of the Poisson process, each LP_i evolves its local time as $LVT_i(t+1) = LVT_i(t) + \eta_i$, and synchronizes with its neighbors at positons $i-1$ and $i+1$ according to the conservative update rules, that have been discussed previously. This means that the LVT_i can only advance if it is a local minimum, i.e. if $\mathrm{LVT}_i(t) \leq min\{\mathrm{LVT}_{i-1}(t), \mathrm{LVT}_{i+1}(t)\}$.

Due to the asynchrony the individual LVTs may decorrelate, which leads to the evolution of a *rough interface* of the VTH. Systems of non-equilibrium surface growth with rough growth exponents are described by the the Kardar-Parisi-Zhang (KPZ) equation, which is given by

$$\frac{\partial \hat{\tau}}{\partial t} = -\gamma \hat{\tau} + \frac{\partial^2 \hat{\tau}}{\partial x^2} - \lambda \left(\frac{\partial \hat{\tau}}{\partial x} \right)^2 + \zeta(x,t), \tag{4.8}$$

where $\hat{\tau}(x,t)$ is the coarse-grained surface height fluctuation measured from the mean resembling the LVT, $\zeta(x,t)$ is uncorrelated Gaussian noise, and coefficients γ and λ carry the details of the coarse-grained growth procedure.

Setting $\tau_i(t) = \mathrm{LVT}_i(t)$ the author shows that (4.8) accurately predicts the efficiency of the PDES for an asymptotically large number of LPs [36] . Ac-

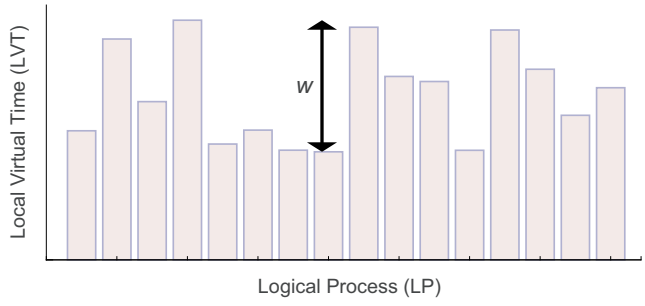

Figure 4.4. The horizon of Virtual Local Times and the statistical spread w on 16 LPs at a given snapshot in real-time.

cording the study, a key parameter influencing the modeled parallel efficiency is the statistical spread of the VTH given as

$$E[w^2(t)] = E\left[\frac{1}{L}\sum_{k=1}^{L}(\tau_k(t) - E_L[\tau(t)]^2\right], \tag{4.9}$$

where $E_L[\tau(t)])$ is the mean spread of the surface at time t over the system size L. For an exemplary VTH the spread is also also visualized in Figure 4.4.

In [36] the authors show that model parametrizations leading to a large value of w scale significantly worse than models with a smaller average spread of the VTH. The spread is typically lower for a model with large system sizes L, as opposed to models with a smaller spatial domains where fluctuations of the VTH are observed when the simulation is carried out with a large number of LPs.

Finally, in [35] the authors demonstrate how adding an additional blocking dependency to each LP active in the simulation has an positive effect of the performance of their simulator. Each LP may than only advance the simulation if its LVT is smaller than the LVTs of the neighbors, as well as the LVT of an other randomly chosen LP. The authors thereby achieve to reduce the spread of the VTH, which leads to a reduction of the total blocking cost in the simulation. This strategy is thus comparable to the hybrid PDES methods discussed in §4.3.1, which aim to equal the LVT growth on all LPs throughout the simulation.

As this approach is unique for the scalability prediction of PDES, these results should be kept in mind when arguing about PDES performance results. However, in which degree the performance model is applicable to prediction for models with a more evolved spatial topology, simulated with, for example, optimistic protocols, is not clear.

4.4 Asynchronous sampling of RDME models

In the following section I will provide some additional details on the implementation of parallel RDME samplers, which have been developed in papers III and IV. In §4.4.1 I discuss the implementations of the correctness criteria in the parallel sampler implementation that enable the convergence of the parallel into the sequential solution.. Finally, in §4.4.2 I discuss some details on the implementation of the Time Window Estimate technique.

4.4.1 Correctness

An important step in the development of asynchronous samplers for stochastic models is to ensure that the implementations of the simulation algorithm is correct. This is a non-trivial task, as the code complexity of the parallel implementations typically rises strongly in comparison to the sequential versions. To achieve a proof of correctness in our implementation of the parallel RDME sampler, we decided to allow for strong converge of the sampler into individual trajectories, thus allowing to generate the exact stochastic path given by the sequential sampler. Note that this is different to other proposed parallel RDME samplers, which converge to the sequential solution only weakly [53], and thus the correctness of the algorithms may only be proven by statistical tests.

The implementation of the in-trajectory convergence is inspired by the Common Reaction Path (CRP) method for sensitivity estimation, discussed in §3.3. The important property of the CRP is that the method dictates to use individual streams of random number for individual stochastic processes. Relevantly for our application, such streams may be also assigned if the processes are distributed over several LPs, which enables the in-trajectory generation to be completely independent on the spatial domain decomposition.

The handling of rollbacks on the other hand is inspired by the technique of *reverse computations*, as used in PDES [7]. This technique prescribes that, in case of a rollback, the opposite operation of the state update is carried out, effectively re-setting the state to the time before the event update. Of relevance to our applications, the generation of random numbers can be reversed too [7], which allows to access the stream of random numbers forwards as well as backwards in time. This technique is not only helpful for implementation of the in-trajectory convergence, but also ensures that *no sampling bias* occurs in parallel execution. Such bias could be a result of, for example, the systematic rejection of events that result from sampling of random numbers that are close to the maximum range of the random variable (e.g. uniformly distributed random numbers close to 1).

The reverse operation of a state update is typically realized by applying the negative of the stoichiometry matrix $\mathbb{N}$. Typically, the necessary information for the reversal of the event, as for example the ID of the reaction or diffusion

event, can be stored and read from the *event history*. The event times for the main events that fired at a given simulation time can be reset by either reverting the seed of the random number generation to the previous sample or acquiring the previous event time from the history list. The time for dependent events can be straight-forwardly recomputed backwards using (3.3).

However, it is very challenging to implement the in-trajectory convergence property for models where the rate w_r of some event r may be set to $w_r = 0$, typically due to a decrease of a dependent copy number to zero. In this case, the original NSM algorithm prescribes to set the next event time $\tau_r = \infty$, i.e., set the transition to be inactive. When the rate of the event is set to $w_r > 0$ at a later point in time, a new next event time τ_r is sampled using Equation (3.1).

In order to allow for the in-trajectory convergence in such situations, we need to enhance the solver with the reversible transitions of processes at zero rate. The computation of the next event time that have to be carried out during forward and backward simulation are noted in Table 4.1 for the main event that occurred at a specific time step or was rollbacked. Likewise, the computation of the next event time for events that are dependent on the main event are noted in Table 4.2. Here we denote w^{t^-} as the event rate *before* the firing time t.

Table 4.1. *Forward and backward computation of the next event time τ_r of a main event r.*

$w_r^{t^-} \to w_r^t$	Forward	Backward
$0 \to 0$	X	X
$0 \to \mathbb{R}_+$	X	X
$\mathbb{R}_+ \to 0$	set to 0	`read from history`
$\mathbb{R}_+ \to \mathbb{R}_+$	$exp(1/\lambda)$	`read from history` & `revert RNG`

Table 4.2. *Forward and backward computation of the next even time τ_r of a dependent event r.*

$w_r^{t^-} \to w_r^t$	Forward	Backward
$0 \to 0$	X	X
$0 \to \mathbb{R}_+$	`retrieve fuse`	`store fuse`
$\mathbb{R}_+ \to 0$	`store fuse`	`retrieve fuse`
$\mathbb{R}_+ \to \mathbb{R}_+$	`rescale`	`rescale`

The operation `read from history` reads the value of τ from the history list, when the event is rollbacked. This information is created by the simulator during every forward processing of an event. The operation `revert RNG` sets the random number generator seed for the corresponding type of event, i.e. it "undoes" the sampling done forward in time.

The operation `rescale` is the evaluation of the Equation (3.3) for dependent event times. The operations `store fuse` and `retrieve fuse` are also used in the sequential implementation of the AEM, and guarantee the Common Reaction Path property for event intensities that are temporally zero. The

operation `store fuse` stores the pre-operational time of event r whose intensity is set to $w_r(\mathbb{X}) = 0$, into variable τ_r^{∞}. The variable computes as

$$\tau_r^{\infty} = (\tau_r^{\text{old}} - t) w_r^{old}, \tag{4.10}$$

where τ_r^{old} is the next event time, and w_r^{old} is the rate of event r, as determined at simulation time t, respectively. The operations concludes with setting the rate $w_r = 0$ and the next event time $\tau_r = \infty$.

The `retrieve fuse` operation is used at a simulation time after the operation `store fuse`, when the intensity of event r is $w_r(\mathbb{X}) > 0$. The operation then computes a new waiting time τ_j^{new} as

$$\tau_r^{\text{new}} = t + \tau_r^{\infty} / w_r^{new}, \tag{4.11}$$

where w_r^{new} is the new (positive) rate of the rth event at the simulation time t.

In order to enable correct usage of `store fuse` and `retrieve fuse` operations in the parallel solver, we further have to store the pre-operational time τ_r^{∞} at any transition of the event rate $\mathbb{R}_+ \to 0$ in the event history. This allows to find the value for τ_j^{∞} for events j, which were stored several continuous time steps before the local simulation time. Then, when a roll-back action is invoked until simulation time $\mathbf{t}$, the *wake up* function finds the particular time $\tau_{j,\mathbf{t}}^{\infty}$ in the history and uses it to update the new operational time τ_j^{new} according to the above equation.

Together, the reversibility of events ensures the convergence of parallel generated trajectories into sequentially generated trajectories. Although the implementation of this property increases the solver complexity, it facilitates the finding of programming errors and ensures correctness.

4.4.2 Synchronization

In this section I shortly discuss two topics regarding the synchronization of asynchronous parallel samplers for RDME models, that have been introduced in papers III and IV. More specifically, I discuss the generation of Time Window Estimates in §4.4.2, and the effect of relaxation of the causality constraint in §4.4.2.

Time Window Estimates

In paper III we propose a new technique for controlling optimism in PDES, called *Dynamic Local Time Window Estimates (DLTWE)*. Each DLTWE assumes that an LP can estimate timestamps of its next outgoing inter-LP events. The LPs than continuously communicate these estimates to its corresponding neighboring LPs, which use the estimates as bounds for advancing their LVT. The technique can thus be seen as an instance of optimism control, which was discussed in §4.3.1 of this thesis.

In pratice, DLTWEs are computed based on outgoing diffusion events that can be found in a scheduled event list, and whose occurence time is sampled already. If no relevant diffusion events for a specific LP are found in the list, the corresponding DLTWE is set to infinity. In this case the receiving LP may proceed with its local simulation without any temporal bound.

In paper IV we propose an approximative version of the DLTWE for the Direct PSNM algorithm, where sampled timetsamps of future inter-LP diffusion events are not available. The TWE (Time Window Estimate) technique proposes to compute the expected time of the next diffusion event from the inter-LP diffusion rates and the current local simulation time.

Let $Z_1 \dots Z_n$ be independent exponentially distributed random variables with rate parameters $\lambda_1 \dots \lambda_n$. Then

$$\min\{Z_1, \dots, Z_n\} \tag{4.12}$$

is also exponentially distributed, with parameter

$$\lambda = \lambda_1 + \dots + \lambda_n. \tag{4.13}$$

Now let $w_1^b \dots w_n^b$ be the rates of the outgoing inter-LP diffusion events over the subdomain boundary at some simulation time t. The minimum of the set is then given as

$$\lambda^d = \sum_{i=1}^{n} w_i^b, \tag{4.14}$$

where λ^b is now the estimator of the next diffusion time over the subdomain boundary, and is send to the corresponding neighbor as the TWE,

$$\tau = \mathrm{LVT} + \lambda^d, \tag{4.15}$$

where LVT is the local virtual time, or the actual continuous time, on the sender LP. The receiving LP than recieves the TWEs $\tau_1 \dots \tau_n$ from all n neighbors, and computes the bound as the geometric mean

$$\sigma = \left(\prod_{i=1}^{n} \tau_i \right)^{\frac{1}{n}} = \sqrt[n]{\tau_1 \tau_2 \cdots \tau_n}. \tag{4.16}$$

The variable σ is then locally used for optimism control together with a scaling factor k that has to be found empirically, as discussed in Paper IV. Hence, the LVT of the receiving LP can be only increased if $LVT < \sigma \cdot k$, otherwise the execution is blocked.

Relaxing the causality constraint

In a preliminary attempt, we aimed to relax the causality constraint of the Direct PNSM simulator, allowing an LP to accept messages whose times are

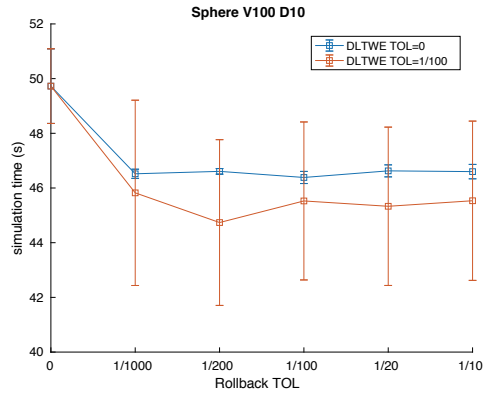

Figure 4.5. Relaxing the causality constraint on the Sphere V100 D10 model, introduced in papers III and IV.

smaller than those of the current LVT, and thus invalidating the local causality constraint, discussed in §4.3. The ultimate goal of the approach is to improve the parallel efficiency by reducing the number of potential rollbacks. In this cases the rate $w_r(\mathbb{X})$ of a diffusion event r would be perturbed with a tolerance ε, resulting in

$$w_r^{la}(\mathbb{X}) = w_r(\mathbb{X})(1+\varepsilon), \tag{4.17}$$

where the perturbed rate w_r^{la} is now higher than the original diffusion intensity $w_r(\mathbb{X})$. The intensity w_r^{la} then leads to the computation of a *last acceptance time*, which is computed as

$$\tau_r^{la} = \exp(1/w_r^{la}(\mathbb{X})), \tag{4.18}$$

and sent together with the diffusion message in case of the diffusion event.

If the LVT of the recieving LP is LVT $> \tau_r$ a rollback would *not be triggered*, as long as LVT $< \tau_r^{la}$. This would lead to an numerical error in the solution of the RDME, which however could be bound in terms of ε, leading to a maximal time at which rollbacks have to be carried out again.

As shown in Figure 4.5, the approach improves the mean efficiency of the parallel simulation only to a small degree, if the constraint was relaxed with a tolerance of $\varepsilon = 1/100$, while the variance of the simulation runtime increases significantly. For higher values of perturbation, surprisingly the performance dropped beyond the baseline of the unperturbed simulation run, mostly due to a strong increase in *blocking time* on the participating LPs.

This result was rather unexpected, as the reduction of rollbacks was assumed to have a positive impact on the performance. However, in regards to §4.3.2, we can suspect that the relaxation of the local causality constraint leads to a growth of a *rough VTH surface*, due to the stronger fluctuation of the LVTs. As discussed in §4.3.2, such fluctuations should be prevented in

asynchronous parallel simulations, as otherwise the blocking time dominates the simulation cost.

5. Summary of Papers

Paper I

P. Bauer, S. Engblom. Sensitivity estimation and inverse problems in spatial stochastic models of chemical kinetics. In *Lecture Notes in Computational Science and Engineering*, 2015.

In this paper we propose a novel method for parameter sensitivity estimation in RDME models. The All Events Method (AEM) is related to the Common Reaction Path method, discussed in §3.3 of this thesis, but can also be applied for sensitivity estimation in spatially extended models. In the paper we compare the efficiency in sensitivity estimation of the AEM and the Next Subvolume Method. Furthermore, we demonstrate how the method can be used for numerical optimization of open-loop control of RDME models.

Contribution
The author of this thesis implemented the method, performed all experiments, wrote and revised the paper in collaboration with SE.

Paper II

P. Bauer, S. Engblom, S. Widgren. Fast event-based epidemiological simulations on national scales. In *The International Journal of High Performance Computing Applications*. 2016.

In this work, we introduce a framework for modeling of infectious disease spread on spatial networks and present an algorithm for parallel simulation of such models on multicores. The modeling framework is also shortly reviewed in §2.2.2 of this thesis. The mathematical model incorporates infectious events as stochastic processes as well as spatial transitions in the form of deterministic events. The parallel simulation is implemented by decomposing event updates into individual tasks and specifying dependencies between them. The parallel simulation is than been carried out via external task libraries `OpenMP` and `SuperGlue`. We show how the method scales well on realistic forward problems as well as on inverse problems and point out parameters that influence the performance.

Contribution
The author of this thesis designed and implemented the parallel method and performed all numerical experiments. The manuscript was written and revised in collaboration of all authors.

Paper III

P. Bauer, J. Lindén, S. Engblom, B. Jonsson. Efficient inter-process synchronization for parallel discrete event simulation on multicores. In *ACM SIGSIM Principles of Advanced Discrete Simulation 2015*.

In this paper we develop a synchronization mechanism for efficient parallel discrete-event simulation of RDME models on multicores. The parallel algorithm is designed on top of the AEM, presented in Paper I. Using the AEM in combination with optimistic parallel discrete-event simulation, we can demonstrate scalable sampling of the RDME. The key aspect of the algorithm is the use of the Dynamic Local Time Window Estimates (DLTWEs), which are estimates of diffusion events that are scheduled ahead of the simulation time and involve diffusion events that are sent from one parallel thread to another. We demonstrate a $\sim 50\%$ parallel efficiency of the parallel method on realistic models and present a regression analysis of the most influential model parameters on the parallel efficiency.

Contribution
The ideas for this work have been developed and implemented in close collaboration between the first and second author. All authors contributed to writing and revising of the paper.

Paper IV

J. Lindén, P. Bauer, S. Engblom, B. Jonsson. Exposing inter-process information for efficient parallel discrete event simulation of spatial stochastic systems. In *ACM SIGSIM Principles of Advanced Discrete Simulation 2017*.

In this paper we show how the synchronization techniques of Paper III can be used for parallelization of the Next Subvolume Method (NSM), a widely used algorithm for simulation of RDME models. More specifically, we present two parallel implementations of the NSM; a Direct Parallel NSM implementation, a straight-forward port to PDES, and the Refined PNSM, which is a hybrid of the parallel AEM method from Paper III and the Next Subvolume Method. The hybrid is demonstrated to keep the scalability of the parallel AEM while reducing the memory overhead of the method in comparison to the NSM. For

the Direct Parallel NSM we also suggest an approximative optimism control, which is shortly reviewed in §4.4.2 of this thesis.

Contribution
The ideas for this work have been developed and implemented in close collaboration between the first and second author. All authors contributed to writing and revising of the paper.

Paper V

P. Bauer, S. Engblom, S. Mikulovic, A. Senek; Multiscale modeling via split-step methods in neural firing. Accepted for publication in *Mathematical and Computer Modeling of Dynamic Systems*, 2017.

In this paper we present a framework for multiscale modeling and simulation of neuronal processes, which is also reviewed in §2.2.3 of this thesis. The model consists of three scales; the gating of ion channels on the neuronal membrane is modeled as a stochastic process at the microscopic scale. At the mesoscopic scale, the dynamics of the neuronal membrane are described as ordinary differential equations, while the macroscopic scale describes the propagation of the transmembrane current into the extracellular space via the Maxwell equations. We present a numerical convergence study of the micro-meso coupling and show examples of the full model coupling.

Contribution
The author of this thesis designed the simulation framework, and performed the numerical experiments together with AS. The manuscript was written and revised in collaboration between all authors.

6. Conclusion

In this thesis I have described applications of event-based modeling in systems biology, epidemiology and neuroscience. I have also provided a discussion of event-based modeling and sequential as well as parallel simulation of such models. In the attached papers I contributed mainly with parallel simulation methodology of event-based models, but also with modeling and simulation techniques, in general.

The main advantages of event-based models are that they allow to include discrete as well as continuous variables in the model state, and that the dynamics can be given by both stochastic and deterministic terms. This allows for high flexibility in the models which in turn may find application in a variety of areas of science and engineering.

Several issues may arise during the simulation of event-based models that contain stochastic terms. As discussed in Paper I, if the aim is to estimate the parameter sensitivity of such models, caution has to be taken in the choice of the simulation method, as otherwise the sensitivity of the model might not be observable due to the fluctuations in the individual samples.

A further topic is how to effectively combine stochastic and deterministic terms in event-based models, where both require a solution in continuous time. In Paper II and V we show that such an approach can often be formulated as a split-step scheme, and that the size of the splitting time step has to be chosen with care, as it may significantly affect the model solution.

As model sizes are growing, or solutions have to be obtained on a large time scale, parallel simulation may be required. Within this thesis I discuss two different possibilities to achieve parallel simulation of single model realizations, which rely on decomposition of the spatial model domain.

One possibility is the usage of synchronous methods, where parallel processors simulate assigned domains during equal time periods, requiring synchronization only between the periods. Although this approach typically allows for greater parallel efficiency, it leads to a discretization error in the solution, if the model is formulated in continuous time.

If an exact solution is preferred, asynchronous parallel simulation must be used, whose implementation is typically significantly more complex than the implementation of synchronous methods. Furthermore, the scalability of such asynchronous simulation is typically very sensitive to the particular implementation as well as to inherent model properties, which makes it hard to predict a priori.

Nonetheless, in Papers III and IV we have decided to pursue this approach, and showed that asychronous simulation of spatial stochastic event-based models can be scalable. Furthermore, we have presented parallel versions of the All Events Method (AEM) and Next Subvolume Method (NSM) algorithms, which will hopefully find use in the Computational Systems Biology community.

For future work, it would be interesting to establish and analyse synchronous parallel simulation of spatial stochastic systems governed by ther reaction-diffusion master equation. More specifically, it would be interesting to implement a closed-loop error control, which allows for adaptive changes of the size of the parallel time interval throughout the simulation, according to an error estimate computed at the previous time step.

Furthermore, more work can be spent into analysing and improving the efficiency of asynchronous simulators. In a preliminary study discussed in the background summary, we have already seen that the relaxation of the causality constraint does not lead to a performance improvement of our asynchronous simulator. It would be interesting to study other possibilities to improve performance, and for example inspect dynamic load balancing of such computations.

Lastly, the modeling frameworks introduced in this thesis may be extended for new applications. It is a great pleasure that the framework introduced in Paper II is very actively used for epidemiological studies at the time of completion of this thesis.

7. Summary in Swedish

Datorsimuleringar och matematiska modeller används ofta för att studera och bättre förstå komplexa förlopp, till exempel olika biologiska processer i celler. Ett flexibelt och vanligt förekommande sätt att modellera sådana processer är att dela upp processen i mindre delsteg eller händelser som kan inträffa i processen och sedan låta slumpen styra när händelserna sker genom att utnyttja datorers förmåga att generera slumptal. Den här typen av matematiska modeller som använder slumpen kallas för *Monte-Carlo* metoder eller stokastiska metoder.

En praktisk svårighet som uppstår vid den här typen av simuleringar är att ju längre tidsförlopp och ju mer komplexa processer som modelleras, desto längre beräkningstid krävs för att genomföra simuleringarna. Den här avhandlingen undersöker metoder för att effektivisera dessa beräkningar genom parallellprogrammering och därmed utnyttja beräkningskraften i flerkärniga moderna datorer. Resultatet blir att den här typen av simuleringar kan genomföras på en bråkdel av tiden.

De specifika beräkningsproblem som studerats i avhandlingen bygger på tillämpningar inom systembiologi, epidemiologi och neurovetenskap. När man studerar dessa system använder man ofta stokastiska modeller som balanserar mellan att beskriva egenskaper på en detaljnivå som tillräckligt noggrannt fångar intressanta och relevanta effekter, men som samtidigt är beräkningsmässigt möjliga att genomföra. Det kan exempelvis handla om att i en stokastisk modell av en kemisk reaktion i en cell, beskriva antalet molekyler av ett protein som finns i cellen men inte exakt hur molekylerna rör sig. I dessa stokastiska modeller är det viktigt att slumpen noggrant återskapar effekter av detaljer som inte ingår i modellen.

I den här avhandlingen presenteras metodik för att med stokastiska modeller simulera den här typen av beräkningsproblem genom ett antal olika implementeringar av parallella händelsebaserade algoritmer. Vilken prestanda som uppnås med de parallella algoritmerna beror ofta kritiskt på specifika detaljer i implementeringen, vilka får stor betydelse. I de flesta av de metoder beskrivs i avhandlingen uppnås en hög beräkningsprestanda som ökar proportionellt med antalet kärnor i datorn som används för simuleringen.

I avhandlingen studeras också vilka specifika egenskaper hos den stokastiska modellen som är viktiga för att uppnå en hög prestandaökning genom parallellisering och vilka egenskaper som förhindrar en optimal parallelliseringen. Dessutom beskrivs två olika parallelliseringar av en simuleringsalgoritm som används inom beräkningsbiologi.

Ett annat bidrag i avhandlingen är känslighetsanalys av parametrar i en stokastisk händelsebaserad modell. Resultaten visar att det är viktigt vilken stokastisk modell som används, eftersom känsligheten hos parameterna eventuellt inte går att uppskatta beroende på variationer mellan körningar av den stokastiska modellen.

Ett annat område som studeras i avhandlingen är tidskontinuerliga modeller som har både deterministiska och stokastiska händelser. I den typen av modeller måste beräkningarna delas upp i mindre tidssteg. Resultaten visar att storleken på dessa tidssteg kan ha stor påverkan på den numeriska lösningen av modellen.

I avhandlingens bakgrund introduceras de olika stokastiska modellerna. Dessutom ges en introduktion till ramverken som används i de olika tillämpningarna. Vidare diskuteras metoder för händelsebaserade simuleringar och känslighetsanalys av stokastiska modeller. Slutligen ges en översikt hur händelsebaserade modeller kan parallelliseras.

En stor del av mjukvaran som utvecklats i den här avhandlingen är tillgänglig som öppen källkod. Eftersom mjukvaran är flexibel, kan den lätt anpassas eller vidareutvecklas för nya frågeställningar och andra tillämpningar.

Acknowledgments

First, I thank my supervisor Stefan Engblom for exchange of ideas, correction of my works, and helping me to become an independent researcher. Secondly, I thank my collaborators Stefan Widgren, Jonatan Lindén and Bengt Jonsson, for doing research together. I have learned a lot from you. I thank UPMARC for the funding of my PhD and for providing a great research infrastructure.

During my studies at TDB, I was lucky to be accompanied by great roommates; thank you Jens, Martin, Magnus, Kristoffer and Siyang for all the fun and support. Furthermore, thanks to all valuable support in research and education from Salman Toor and Andreas Hellander. The journey into my PhD started at the Applied Neurodynamics Lab at BMC. I thank Richardson Leão for the warm welcome in Uppsala and becoming a friend of our family.

I also want to thank my Uppsala friends for your support and encouragement. Fabio and Hejdur, you are our base in this town, thanks for your friendship and all the great times. Behrang and Tina, thank you for being always there, we are very happy to have friends like you. Kali, Sulena and Aura, thanks for always being great family friends, I hope we stay in touch wherever the wind will take us. Andreas, Ida and Alfred as well as Viktor, Inna and Daniel, thanks for the great times before and after we got our little ones. Big thanks also to the rest of the BMC gang, Stefano & Edda, Helton & Rosanna, Kasia & Adriano, Sharn & Marcus, I enjoyed spending time with all of you! Furthermore, I want to thank my old friends Manfred, Olli, and Michi, for keeping in touch no matter how far we are from each other.

From the deepest of my heart I have to thank my family for all the help and support that I received from you. Mama and Papa, you did incredibly much for me and I am afraid I never showed how thankful I am for that. Mama, it leaves me in sorrow that you can not read this lines and hold this book in your hands. I will forever miss you. Papa, thank you for all the support, especially in the difficult times of this year. We are all incredibly happy to have you. Gabi, without your help this thesis could not have been written. Thank you for your support and being a great big sister, I am looking forward for the good times to spend with our families. Mama Mila i Tata Vita, hvala vama za vasu pomoc,vi iste moji drugi roditelji!

Sanja, the love of my life, thank you for making me the one that I am today. We were once best friends and now we are soulmates planting our own seeds of life. I consider myself incredibly lucky to have you and enjoy every move that we make together. Lea and Andrej, you are the sunshine of my life. I don't want to miss any day without you. How incredibly lucky I am to have all of you in my life!

References

[1] Marco Aldinucci, Massimo Torquati, Concetto Spampinato, Maurizio Drocco, Claudia Misale, Cristina Calcagno, and Mario Coppo. Parallel stochastic systems biology in the cloud. *Briefings in Bioinformatics*, 15(5):798–813, 2014.

[2] David F. Anderson. An efficient finite difference method for parameter sensitivities of continuous time markov chains. *SIAM Journal on Numerical Analysis*, 50(5):2237–2258, 2012.

[3] Giorgos Arampatzis, Markos A. Katsoulakis, Petr Plechac, Michela Taufer, and Lifan Xu. Hierarchical fractional-step approximations and parallel kinetic Monte Carlo algorithms. *Journal of Computational Physics*, 231(23):7795–7814, 2012.

[4] Peter D. Barnes, Jr., Christopher D. Carothers, David R. Jefferson, and Justin M. LaPre. Warp Speed: Executing Time Warp on 1,966,080 Cores. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '13, pages 327–336, 2013.

[5] Pavol Bauer. Parallelism and efficiency in discrete-event simulation. *Licentiate Thesis*, 2015.

[6] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of Chemical Physics*, 124(4):044109, 2006.

[7] Christopher D. Carothers, Kalyan S. Perumalla, and Richard M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. *ACM Transactions on Modeling and Computer Simulation*, 9(3):224–253, 1999.

[8] Christos G. Cassandras and Stéphane Lafortune. *Systems and Models*, pages 1–51. Springer US, 2008.

[9] K. M. Chandy and J. Misra. Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Commun. ACM*, 24(4):198–206, 1981.

[10] S. R. Das. Adaptive protocols for parallel discrete event simulation. *The Journal of the Operational Research Society*, 51(4):385–394, 2000.

[11] Lorenzo Dematté and Tommaso Mazza. On parallel stochastic simulation of diffusive systems. In *Computational Methods in Systems Biology*, number 5307 in Lecture Notes in Computer Science, pages 191–210. Springer Berlin Heidelberg, 2008.

[12] B. Drawert, M. Trogdon, S. Toor, L. Petzold, and A. Hellander. MOLNs: A Cloud Platform for Interactive, Reproducible, and Scalable Spatial Stochastic Computational Experiments in Systems Biology Using PyURDME. *SIAM Journal on Scientific Computing*, 38(3):C179–C202, 2016.

[13] Brian Drawert, Michael J. Lawson, Linda Petzold, and Mustafa Khammash. The diffusive finite state projection algorithm for efficient simulation of the stochastic reaction-diffusion master equation. *The Journal of Chemical Physics*, 132(7):074101, 2010.

[14] J Elf and M Ehren. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems biology*, 1(2):230–236, 2004.
[15] S. Engblom. Parallel in Time Simulation of Multiscale Stochastic Chemical Kinetics. *Multiscale Modeling & Simulation*, 8(1):46–68, 2009.
[16] Stefan Engblom. On the stability of stochastic jump kinetics. *Applied Mathematics*, 51:3217–3239, 2014.
[17] Stefan Engblom, Lars Ferm, Andreas Hellander, and Per Lötstedt. Simulation of stochastic reaction-diffusion processes on unstructured meshes. *SIAM Journal on Scientific Computing*, 31(3):1774–1797, 2009.
[18] S. N. Ethier and T. G. Kurtz. *Markov Processes: Characterization and Convergence*. Wiley series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1986.
[19] Richard M. Fujimoto. *Parallel and Distribution Simulation Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.
[20] Richard M. Fujimoto. Research Challenges in Parallel and Distributed Simulation. *ACM Trans. Model. Comput. Simul.*, 26(4):22:1–22:29, 2016.
[21] Crispin W. Gardiner. *Handbook of stochastic methods for physics, chemistry, and the natural sciences*. Springer, 1985.
[22] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.
[23] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
[24] Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Computational Biology*, 3(10):e189, 2007.
[25] Georg Hager and Gerhard Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2010.
[26] Andreas Hellander, Michael J. Lawson, Brian Drawert, and Linda Petzold. Local error estimates for adaptive simulation of the reaction diffusion master equation via operator splitting. *Journal of Computational Physics*, 266:89–100, 2014.
[27] A L Hodgkin and A F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952.
[28] Krishna A. Iyengar, Leonard A. Harris, and Paulette Clancy. Accurate implementation of leaping in space: The spatial partitioned-leaping algorithm. *The Journal of Chemical Physics*, 132(9):094101, 2010.
[29] David R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, 1985.
[30] Matthias Jeschke, Roland Ewald, Alfred Park, Richard Fujimoto, and Adelinde M. Uhrmacher. A parallel and distributed discrete event approach for spatial cell-biological simulations. *SIGMETRICS Perform. Eval. Rev.*, 35(4):22–31, 2008.
[31] Matthias Jeschke, Roland Ewald, and Adelinde M. Uhrmacher. Exploring the

performance of spatial stochastic simulation algorithms. *Journal of Computational Physics*, 230(7):2562–2574, 2011.

[32] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, August 2004.

[33] W. O. Kermack and A. G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 115(772):700–721, 1927.

[34] A. Kolakowska, M. A. Novotny, and Per Arne Rikvold. Update statistics in conservative parallel-discrete-event simulations of asynchronous systems. *Physical Review E*, 68(4):046705, 2003.

[35] G. Korniss, M. A. Novotny, H. Guclu, Z. Toroczkai, and P. A. Rikvold. Suppressing roughness of virtual times in parallel discrete-event simulations. *Science (New York, N.Y.)*, 299(5607):677–679, 2003.

[36] G. Korniss, M. A. Novotny, A. K. Kolakowska, and H. Guclu. Statistical Properties of the Simulated Time Horizon in Conservative Parallel Discrete-event Simulations. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 132–137, 2002.

[37] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. A "parareal" in time discretization of PDEs. *Comptes Rendus de l 'academie des Sciences - Series I - Mathematics*, 332(7):661–668, 2001.

[38] Jason Liu. *Parallel Discrete-Event Simulation*. John Wiley & Sons, Inc., 2010.

[39] Jason Liu and David M. Nicol. Lookahead revisited in wireless network simulations. In *Proceedings of 16th Workshop on Parallel and Distributed Simulation*, pages 79–88. IEEE, 2002.

[40] A.W. Malik, A. Park, and R.M. Fujimoto. Optimistic Synchronization of Parallel Simulations in Cloud Computing Environments. In *IEEE International Conference on Cloud Computing, 2009. CLOUD '09*, pages 49–56, 2009.

[41] Tommaso Mazza, Paolo Ballarini, Rosita Guido, and Davide Prandi. The Relevance of Topology in Parallel Simulation of Biological Networks. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 9(3):911–923, 2012.

[42] A. Moraes, R. Tempone, and P. Vilanova. Hybrid Chernoff Tau-Leap. *Multiscale Modeling & Simulation*, 12(2):581–615, 2014.

[43] James Nutaro. *Discrete event simulation of continuous systems*. 2005.

[44] Yannis Pantazis, Markos A. Katsoulakis, and Dionisios G. Vlachos. Parametric sensitivity analysis for biochemical reaction networks based on pathwise information theory. *BMC bioinformatics*, 14:311, 2013.

[45] Johan Paulsson, Otto G. Berg, and M. Ehrenberg. Stochastic focusing: Fluctuation-enhanced sensitivity of intracellular regulation. *Proceedings of the National Academy of Sciences*, 97(13):7148–7153, June 2000.

[46] Sergey Plyasunov and Adam P. Arkin. Efficient stochastic sensitivity analysis of discrete event systems. *Journal of Computational Physics*, 221(2):724–738, 2007.

[47] Muruhan Rathinam, Patrick W. Sheppard, and Mustafa Khammash. Efficient computation of parameter sensitivities of discrete stochastic chemical reaction networks. *The Journal of Chemical Physics*, 132(3):034103, 2010.

[48] J. Vidal Rodríguez, Jaap A. Kaandorp, Maciej Dobrzyński, and Joke G. Blom. Spatial stochastic modelling of the phosphoenolpyruvate-dependent

phosphotransferase (PTS) pathway in escherichia coli. *Bioinformatics*, 22(15):1895–1901, 2006.

[49] Sudhir Srinivasan and Paul F. Reynolds, Jr. Elastic Time. *ACM Trans. Model. Comput. Simul.*, 8(2):103–139, 1998.

[50] Rishi Srivastava, David F. Anderson, and James B. Rawlings. Comparison of finite difference based methods to obtain sensitivities of stochastic chemical kinetic models. *The Journal of Chemical Physics*, 138(7):074110, 2013.

[51] Jeff S. Steinman. Discrete-event Simulation and the Event Horizon. In *Proceedings of the Eighth Workshop on Parallel and Distributed Simulation*, PADS '94, pages 39–49, 1994.

[52] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[53] Bing Wang, Bonan Hou, Fei Xing, and Yiping Yao. Abstract next subvolume method: A logical process-based approach for spatial stochastic simulation of chemical reactions. *Comput. Biol. Chem.*, 35(3):193–198, 2011.

[54] Jingjing Wang, Deepak Jagtap, Nael B. Abu-Ghazaleh, and Dmitry Ponomarev. Parallel discrete event simulation for multi-core systems: Analysis and optimization. *IEEE Trans. Parallel Distrib. Syst.*, 25(6):1574–1584, 2014.

[55] Barry Williams, Dmitry Ponomarev, Nael Abu-Ghazaleh, and Philip Wilsey. Performance Characterization of Parallel Discrete Event Simulation on Knights Landing Processor. In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '17, pages 121–132, 2017.

[56] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary. Scheduling Critical Channels in Conservative Parallel Discrete Event Simulation. In *In Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, pages 20–28, 1999.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 1586

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-332009

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2017